Amendments to the Claims:

This listing of Claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. – 16. (Canceled)

17. (Currently Amended) A method for updating program code stored in a memory, the memory having a plurality of memory sectors, the method comprising the steps of:

transforming at least one updated source code module into an updated program code version to be stored in a memory, which memory has stored thereon a current program code version occupying a first set of the memory sectors of the memory, wherein the updated program code version occupies a second set of memory sectors when stored in the memory;

wherein the transforming step further comprises the steps of

compiling the at least one source code module resulting in a number of object modules,

receiving a representation of the current program code version, ~~and~~

performing at least one optimization step adapted to decrease the number of memory sectors of the second set of memory sectors occupied by the updated code version that are different from the corresponding memory sectors of the first set of memory sectors occupied by the current program code version, and

controlling the at least one optimization step by at least one optimization parameter, wherein the at least one optimization parameter includes a maximum bound on allowed padding space that a linker is allowed to introduce or a maximum number of relays that the linker is allowed to introduce; and

wherein the performing at least one optimization step, further comprises the steps of generating feedback data during a linking step for linking the number of object modules, re-compiling at least a subset of the source code modules based on the

feedback data and resulting in a number of modified object modules, and performing a linking step based on the number of modified object modules.

18. (Previously Presented) The method according to claim 17, wherein the representation of the current program code version comprises at least one of a current image of the first set of memory sectors and a map file description of the current image of the first set of memory sectors.

19. (Cancelled)

20. (Previously Presented) The method according to claim 17, wherein the optimization step further comprises the step of determining a layout of the object modules in memory.

21. (Previously Presented) The method according to claim 20, wherein determining the layout of the object modules in memory further comprises the steps of:

detecting an updated first object module having a different size than a corresponding first current object module, and an updated second object module equal to a corresponding second current object module, which updated second object module has a base address larger than the base address of the updated first object module; and

padding the detected updated first object module with a predetermined memory content of a predetermined padding size resulting in a padded updated first object module; wherein the padding size is selected to cause the base address of the updated second object module to be equal to the base address of the corresponding second current object module.

22. (Previously Presented) The method according to claim 20, wherein determining the layout of the object modules in memory further comprises the steps of:

detecting an updated first object module that is larger than a corresponding first current object module;

moving a predetermined part of the updated first object module to a different memory sector resulting in a reduced updated first object module and a moved part of the updated first object module; and

inserting a reference to the moved part of the updated first object module in the reduced first updated memory sector.

23. (Cancelled)

24. (Currently Amended) The method according to claim [[23]]17, wherein the at least one optimization parameter includes a parameter determining a maximum allowed increase in size caused by the optimization step.

25. (Currently Amended) The method according to claim [[23]]17, wherein the at least one optimization parameter includes a parameter determining a maximum allowed number of references introduced by the optimization step.

26. (Previously Presented) The method according to claim 17, further comprising generating a delta file representative of differences between the current program code version and the updated program code version.

27. (Previously Presented) The method according to claim 17, wherein the memory is a flash memory.

28. (Previously Presented) The method according to claim 17, wherein the memory is a memory of a portable radio communications equipment.

29. (Currently Amended) A data processing system for updating program code stored in a memory, the memory having a plurality of memory sectors, the data processing system comprising:

transformation means adapted to transform at least one updated source code module into an updated program code version to be stored in a memory, which memory

has stored thereon a current program code version occupying a first set of the memory sectors of the memory, wherein the updated program code version occupies a second set of memory sectors when stored in the memory;

the transformation means further having

a compilation means adapted to compile the at least one source code module resulting in a number of object modules,

a reception means adapted to receive a representation of the current program code version, and

a performance means adapted to perform at least one optimization operation to decrease the number of memory sectors of the second set of memory sectors occupied by the updated code version that are different from the corresponding memory sectors of the first set of memory sectors occupied by the current program code version; and

wherein the performance means adapted to perform the at least one optimization operation is further adapted to generate feedback data while the number of object modules are being linked, re-compile at least a subset of the source code modules based on the feedback data resulting in a number of modified object modules, and perform a linking operation based on the number of modified object modules, wherein the at least one optimization operation is controlled by at least one optimization parameter and the at least one optimization parameter includes a maximum bound on allowed padding space that a linker is allowed to introduce or a maximum number of relays that the linker is allowed to introduce.

30. (Previously Presented) The data processing system according to claim 29, wherein the representation of the current program code version comprises at least one of a current image of the first set of memory sectors and a map file description of the current image of the first set of memory sectors.

31. (Cancelled)

32. (Previously Presented) The data processing system according to claim 29, wherein the optimization operation is adapted to determine a layout of the object modules in memory.

33. (Previously Presented) The data processing system according to claim 32, wherein the determination of the layout of the object modules in memory further comprises:

a detection operation adapted to detect an updated first object module having a different size than a corresponding first current object module, and an updated second object module equal to a corresponding second current object module, which updated second object module has a base address larger than the base address of the updated first object module; and

a padding operation adapted to pad the detected updated first object module with a predetermined memory content of a predetermined padding size resulting in a padded updated first object module; wherein the padding size is selected to cause the base address of the updated second object module to be equal to the base address of the corresponding second current object module.

34. (Previously Presented) The data processing system according to claim 32, wherein the determination of the layout of the object modules in memory further comprise:

a detection operation adapted to detect an updated first object module that is larger than a corresponding first current object module;

a movement operation adapted to move a predetermined part of the updated first object module to a different memory sector resulting in a reduced updated first object module and a moved part of the updated first object module; and

an insertion operation adapted to insert a reference to the moved part of the updated first object module in the reduced first updated memory sector.

35. (Previously Presented) The data processing system according to claim 29, wherein the transformation operation is further adapted to control the optimization operation by at least one optimization parameter.

36. (Previously Presented) The data processing system according to claim 35, wherein the at least one optimization parameter includes a parameter determining a maximum allowed increase in size caused by the optimization step.

37. (Previously Presented) The data processing system according to claim 35, wherein the at least one optimization parameter includes a parameter determining a maximum allowed number of references introduced by the optimization step.

38. (Previously Presented) The data processing system according to claim 29, further comprising a generation operation adapted to generate a delta file representative of differences between the current program code version and the updated program code version.

39. (Previously Presented) The data processing system according to claim 29, wherein the memory is a flash memory.

40. (Previously Presented) The data processing system according to claim 29, wherein the memory is a memory of a portable radio communications equipment.

41. (Currently Amended) A computer program product comprising program code means adapted to cause a data processing system to perform steps when the program code means are executed on the data processing system, the steps comprising:

transforming at least one updated source code module into an updated program code version to be stored in a memory, which memory has stored thereon a current program code version occupying a first set of the memory sectors of the memory,

wherein the updated program code version occupies a second set of memory sectors when stored in the memory;

wherein the transforming step further comprises the steps of

compiling the at least one source code module resulting in a number of object modules,

receiving a representation of the current program code version, ~~and~~

performing at least one optimization step adapted to decrease the number of memory sectors of the second set of memory sectors occupied by the updated code version that are different from the corresponding memory sectors of the first set of memory sectors occupied by the current program code version, and

controlling the at least one optimization step by at least one optimization parameter, wherein the at least one optimization parameter includes a maximum bound on allowed padding space that a linker is allowed to introduce or a maximum number of relays that the linker is allowed to introduce; and

wherein performing the at least one optimization step, further comprises the steps of generating feedback data during a linking step for linking the number of object modules, re-compiling at least a subset of the source code modules based on the feedback data and resulting in a number of modified object modules, and performing a linking step based on the number of modified object modules.

42. (Previously Presented) The computer program product according to claim 41, wherein the computer program product comprises a linker module.